Machine Readable Travel Documents (MRTD) support in uFCoder library

Document version 1.0

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Table of contents

Introduction

Support for reading data groups from NFC tag embedded in the Machine Readable Travel Documents (MRTDs), including ePassports that comply with ICAO specifications, has been implemented in the uFCoder library.

Implementation supports Basic Access Control (BAC) mechanism for NFC chip access. BAC enable authentication and secure cryptographic communication channel with an NFC tag embedded in the MRTD. BAC is based purely on symmetric cryptography using 3DES algorithm and it is implemented according to ICAO 9303, part 11.

ICAO stands for International Civil Aviation Organization (https://www.icao.int). ICAO 9303 specification standardizes MRTDs, including ePassports. You can find entire ICAO Doc 9303 series on https://www.icao.int/publications/pages/publication.aspx?docnum=9303 web location.

MRTDs Basic Access Control is supported in **uFCoder library** from **version 5.0.12**.

In order to authenticate to NFC tag embedded in the MRTD first you have to pass document number, birth date of the document holder and document expiration date to function MRTD_MRZDataToMRZProtoKey() in order to get "proto key" from which will be derived other necessary security keys. All the data needed to get "proto key" (document number, birth date of the document holder and document expiration date) are encoded in Machine Readable Zone (MRZ) so the library have MRTD_MRZSubjacentToMRZProtoKey() function which can be called instead of MRTD_MRZDataToMRZProtoKey(). This function accept null terminated string containing subjacent row of the document MRZ. On the figure 1 you can see an example of the MRZ with marked subjacent row which content you have to pass as parameter to function MRTD_MRZSubjacentToMRZProtoKey().

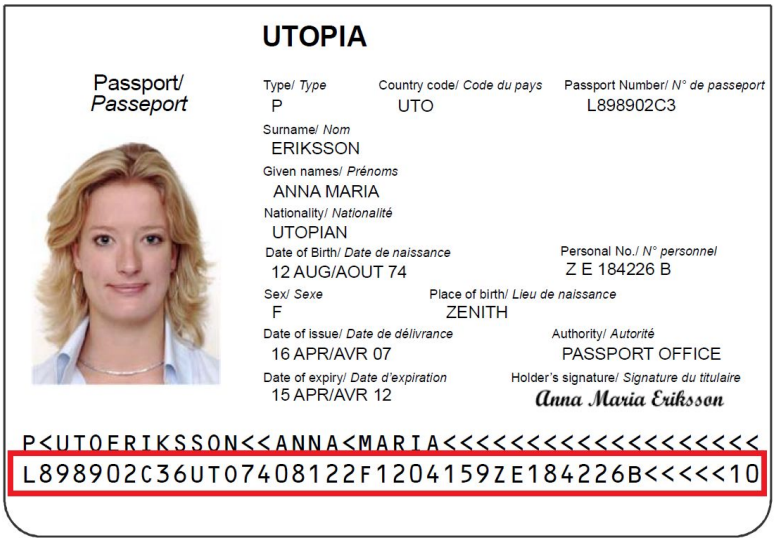

figure 1

Library functions for MRTD support

MRTD_MRZDataToMRZProtoKey

Function description

In order to get MRZ Proto Key needed in subsequent steps, you can call this function and pass it null terminated strings containing document number, document holder date of birth and document expiration date. After successful function execution MRZ Proto Key will be stored in a mrz_proto_key 25-byte array.

Function declaration (C language)

```
UFR_STATUS MRTD_MRZDataToMRZProtoKey(const char *doc_number,
                                     const char *date_of_birth,
                                     const char *date_of_expiry,
                                     uint8_t mrz_proto_key[25]);
```

Parameters

doc_number	Pointer to a null terminated string containing exactly 9 characters document number.
date_of_birth	Pointer to a null terminated string containing exactly 6 characters representing the date of birth in the "YYMMDD" format.
date_of_expiry	Pointer to a null terminated string containing exactly 6 characters representing expiration date in the "YYMMDD" format.
mrz_proto_key	This byte array will contain calculated MRZ proto-key after successful function execution. This array must have allocated at least 25 bytes prior calling this function.

MRTD_MRZSubjacentToMRZProtoKey

Function description

In order to get MRZ Proto Key needed in subsequent steps, in case of the TD3 MRZ format (88 totally character long), you can call this function and pass it null terminated string containing MRZ subjacent row. Example of the TD3 MRZ format printed on the eMRTD document looks like this:

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

```
P<UTOERIKSSON<<ANNA<MARIA<<<<<<<<<<<<<<<<<<<
L898902C36UTO7408122F1204159ZE184226B<<<<<10
```

This function should receive a pointer to a null terminated string containing MRZ subjacent row i.e. "L898902C36UTO7408122F1204159ZE184226B<<<<<10".

Function declaration (C language)

```
UFR_STATUS MRTD_MRZSubjacentToMRZProtoKey(const char *mrz,
                                          uint8_t mrz_proto_key[25]);
```

Parameters

mrz	Pointer to a null terminated string containing MRZ data. According to ICAO Doc 9303-10, there it has three MRZ data formats: TD1,TD2 or TD3 formats. TD1 contains exactly 90 characters, TD2 contains exactly 72 characters and TD3 contains exactly 88 characters.
mrz_proto_key	This byte array will contain calculated MRZ proto-key after successful function execution. This array must have allocated at least 25 bytes prior calling this function.

MRTDAppSelectAndAuthenticateBac

Function description

Use this function to authenticate to the eMRTD NFC tag using BAC. This function establish security channel for communication. Security channel is maintained using send_sequence_cnt parameter and channel session keys are ksenc (for encryption) and ksmac (for calculating MAC).

Function declaration (C language)

```
UFR_STATUS MRTDAppSelectAndAuthenticateBac(const uint8_t mrz_proto_key[25],
                                           uint8_t ksenc[16],
                                           uint8_t ksmac[16],
                                           uint64_t *send_sequence_cnt);
```

Parameters

mrz_proto_key	MRZ proto-key acquired using prior call to MRTD_MRZDataToMRZProtoKey() or MRTD_MRZSubjacentToMRZProtoKey() function.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

ksenc	This array must have allocated at least 16 bytes prior calling this function. This array will contain session encryption key after successful function execution.
ksmac	This array must have allocated at least 16 bytes prior calling this function. This array will contain session key for calculating MAC after successful function execution.
send_sequence_cnt	After successful execution of this function, pointer to this 64-bit value should be saved and forwarded at every subsequent call to MRTDFileReadBacToHeap() and/or other functions for reading eMRTD.

MRTDFileReadBacToHeap

Function description

Use this function to read files from the eMRTD NFC tag. You can call this function only after successfully established security channel by the previously called MRTDAppSelectAndAuthenticateBac() function. Session keys ksenc and ksmac, and also parameter send_sequence_cnt are acquired by the previously called MRTDAppSelectAndAuthenticateBac() function. After the successful call to this function, *output points to the file data read from a eMRTD file specified by the file_index parameter. Buffer, in which the data is stored, is automatically allocated on memory heap during function execution. Maximum amount of data allocated can be 32KB. There is programmer responsibility to cleanup allocated data (i.e. by calling free(), the standard C function) after use.

Function declaration (C language)

```
UFR_STATUS MRTDFileReadBacToHeap(const uint8_t *file_index,
                                 uint8_t **output,
                                 uint32_t *output_length,
                                 const uint8_t ksenc[16],
                                 const uint8_t ksmac[16],
                                 uint64_t *send_sequence_cnt);
```

Parameters

file_index	Parameter that specifies the file we want to read from the eMRTD. This is pointer to byte array contains exactly two bytes designating eMRTD file. Those two bytes are file identificator (FID) and there is a list of FIDs: EF.COM = {0x01, 0x1E}

	EF.DG1 = {0x01, 0x01} EF.DG2 = {0x01, 0x02} EF.DG3 = {0x01, 0x03} EF.DG4 = {0x01, 0x04} EF.DG5 = {0x01, 0x05} EF.DG6 = {0x01, 0x06} EF.DG7 = {0x01, 0x07} EF.DG8 = {0x01, 0x08} EF.DG9 = {0x01, 0x09} EF.DG10 = {0x01, 0x0A} EF.DG11 = {0x01, 0x0B} EF.DG12 = {0x01, 0x0C} EF.DG13 = {0x01, 0x0D} EF.DG14 = {0x01, 0x0E} EF.DG15 = {0x01, 0x0F} EF.DG16 = {0x01, 0x10} EF.SOD = {0x01, 0x1D}
***output**	After the successful call to this function, this pointer points to the file data read from a eMRTD file specified by the file_index parameter. Buffer, in which the data is stored, is automatically allocated during function execution. Maximum amount of data allocated can be 32KB. There is programmer responsibility to cleanup allocated data (i.e. by calling free(), the standard C function) after use.
output_length	After the successful call to this function, this pointer is points to the size of the file data read from a eMRTD file specified by the file_index parameter.
ksenc	Session encryption key acquired using prior call to MRTDAppSelectAndAuthenticateBac() function.
ksmac	Session key for calculating MAC acquired using prior call to MRTDAppSelectAndAuthenticateBac() function.
send_sequence_cnt	This pointer should point to a 64-bit value initialized by the previously successful call to MRTDAppSelectAndAuthenticateBac() function. Pointer to this 64-bit value should be saved and forwarded at every subsequent call to this function and/or other functions used for reading eMRTD.

ePassport MRTD Example

This example you can download from:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-examples-ePassport_mrtd.git
or clone the entire eclipse cdt project using:
`git clone --recursive https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-examples-ePassport_mrtd.git`
command.
If you want quick run only, download the project and start binary executable from the appropriate folder:

- for a 32-bit Windows start the win32_release\run_me.cmd
- for a 64-bit Windows start the win64_release\run_me.cmd
- for a 32-bit Linux start linux32_release/ePassport_mrtd
- for a 64-bit Linux start linux64_release/ePassport_mrtd

Software example requires uFR reader device to be attached and configured to the PC. No other application or service using uFR reader should be running on the computer. After successful start of the "ePassport MRTD Example" you will get screen like on figure 2.

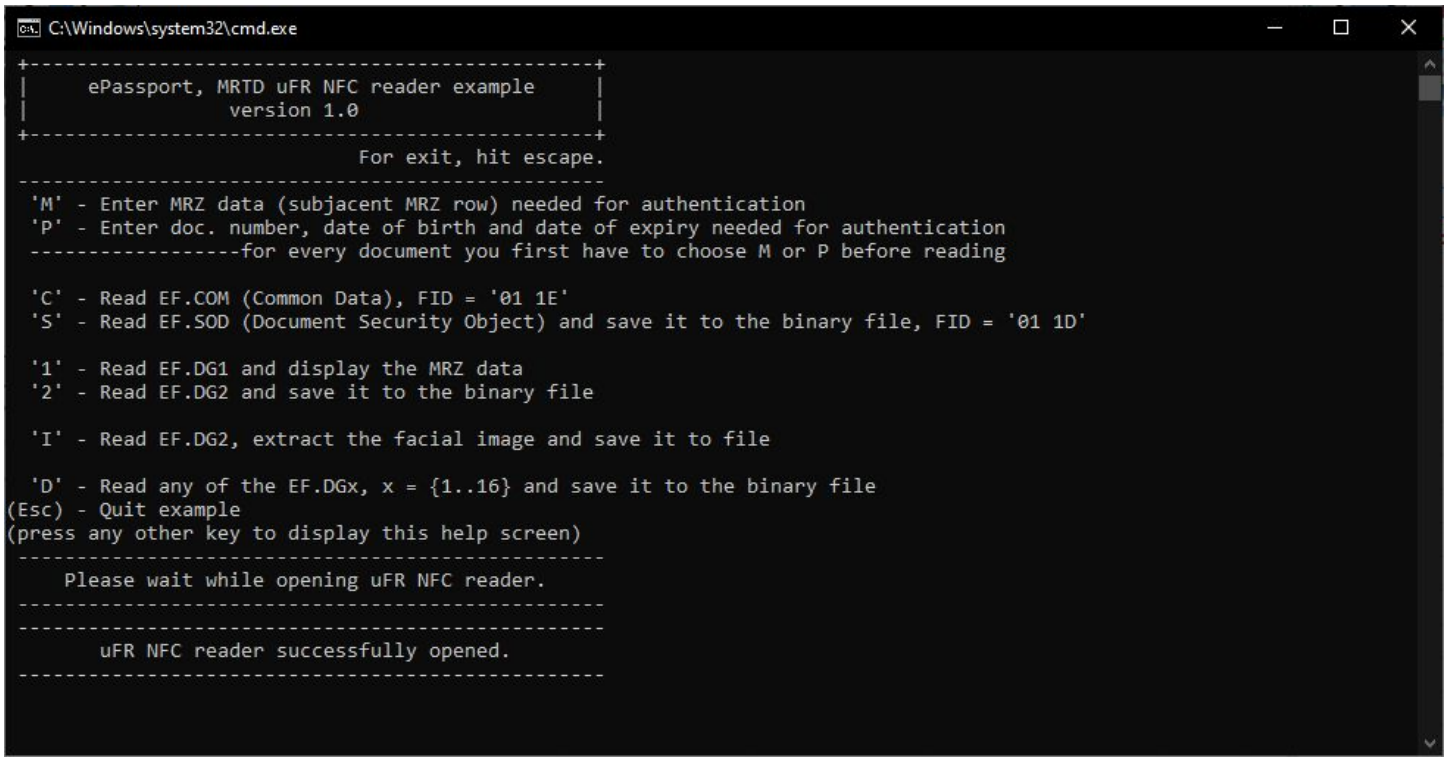

figure 2: "ePassport MRTD Example" start screen

Now, you should choose one of the 'M' or 'P' options as stated in the application usage instructions on the screen.
If you chose 'M' option, you will be prompted with text:

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

```
You have chose to enter subjacent MRZ row located under the
'P<XXXSURNAME<<FIRSTNAME<<<<<<<<<<<<<<<<<<<<':

Enter subjacent MRZ row. Subjacent MRZ row have to be 44 characters long.
```

so enter subjacent MRZ row. Example of the subjacent MRZ row you can see on the figure 1.

Otherwise, if you chose 'P' option you will be prompted with text:

```
You have chose to enter doc. number, date of birth and date of expiry
separately:

Enter the document number. The document number should be 9 characters long.
_________ ...
Enter date of birth. Date format have to be YYMMDD.
______ ...
Enter date of expiry. Date format have to be YYMMDD.
______ ...
```

so enter the data in the appropriate format.

After you have entered the valid data, application will inform you with a message:

```
MRZ proto-key has been set successfully.
-----------------------------------------------------------------
```

After this message you can continue with a read operations on the NFC tag embedded in to ePassport that data you have previously entered belongs.

Now you can put ePasspoert in the uFR reader field. On successful communication established you will get basic information about NFC tag in the reader field. For example:

```
-----------------------------------------------------------------
Tag type: DL_GENERIC_ISO14443_4, sak = 0x??, uid[4] = ??:??:??:??
-----------------------------------------------------------------
```

SAK and UID in this example are masked and they can have any arbitrary value. ePassports will be always be recognized like DL_GENERIC_ISO14443_4 tag type.

Now you can chose application reading options:

'C' - this option reads common data (EF.COM elementary file) from the ePassport. After successful reading, data is parsed and displayed in the following format:

```
EF.COM has been successfully read. File length is ?? bytes
Raw data: 60 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx ...
Parsing the EF.COM raw data:
LDS version is 01.07
UNICODE version is 04.00.00
Existing data groups list:
   Found: EF.DG1
   Found: EF.DG2
   Found: EF.DG3
   Found: EF.DG14

-----------------------------------------------------------------
```

Raw data in this example are masked and they can have any arbitrary value. Only the raw data tag has been present and it will be always the same (0x60). When you read your own document, you will get its actual raw data here. More about LDS version and UNICODE version you can read in the ICAO 9303, part 10 document.
LDS and UNICODE version are followed by the data groups list that ePassport contains. Only DG1 and DG2 are mandatory. All the other data groups can be either present or not in the particular MRTD.

'S' - this option reads the document security object ($EF.SO_D$ elementary file) and save it to the binary file which path and name you have to enter when you prompted. Document security object contains digital signature in the standard PKCS#7 CMS format. Presence of the $EF.SO_D$ on the MRTD is mandatory.

'1' - this option reads the EF.DG1, parse it and displays raw and parsed data in the following format:

```
EF.DG1 has been successfully read. File length is ?? bytes
Raw data:
61 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx ...
Simple parsing the EF.DG1 raw data:
   Document code: P  (ePassport)
   Issuing State or organization: ???
   Name of holder: SURNAME  FIRST_NAME
   Document number: ?????????
   Nationality: ???
   Date of birth (dd.MM.yyyy.): ??.??.????.
   Sex: ????
   Date of expiry (dd.MM.yyyy.): ??.??.????.
   Optional data: ????????????????
-----------------------------------------------------------------
```

Raw data in this example are masked and they can have any arbitrary value. Only the raw data tag has been present and it will be always the same (0x61). When you read your own document, you will get its actual raw data here.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

'2' - this option reads the EF.DG2 and save it to the binary file which path and name you have to enter when you prompted. EF.DG2 contains document holder facial image and it is mandatory. EF.DG2 beside facial image could contain biometric facial features to. More about EF.DG2 content you can read in the ICAO 9303, part 10 document.

'I' - this option reads the EF.DG2 to. In this case only the facial image is extracted from the MRTD file and saved to the file which path and name you have entered. Image format is automatically detected and the file extension is set according to it. There are two possible image file formats defined for this context: JPEG or JP2 (i.e. jpeg 2000).

'D' - this option reads any of the elementary data group (EF.DG) files from the MRTD and save it to the binary file which path and name you have to enter when you prompted. After this option has been chosen you will be prompted for EF.DG index. Index can be from the range 1 to 16 (e.g. 1 for EF.DG1 and 14 for EF.DG14). Elementary file you wanted to read must be listed in the EF.COM data groups list.

Reading of some optional elementary files, especially those containing biometric data, requires special security mechanisms which is outside the scope of this document.

Current version of the "ePassport MRTD Example" is 1.0 an depands on the uFCoder library version 5.0.12 and uFR firmware version 5.0.22.

Revision history

Date	Version	Comment
2019-09-19	1.0	Base document

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com